Target Cuts from Relaxed Decision Diagrams

Christian Tjandraatmadja[1], Willem-Jan van Hoeve[1]

[1]Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA
{ctjandra,vanhoeve}@andrew.cmu.edu

Abstract

The most common approach to generate cuts in integer programming is to derive them from the linear programming relaxation. We study an alternative approach that extracts cuts from discrete relaxations known as relaxed decision diagrams. Through a connection between decision diagrams and polarity, the algorithm generates cuts that are facet-defining for the convex hull of a decision diagram relaxation. As proof of concept, we provide computational evidence that this algorithm generates strong cuts for the maximum independent set problem and the minimum set covering problem.

1 Introduction

Most general-purpose cutting planes in integer programming are based on a linear programming (LP) relaxation. However, there may be useful cuts that are difficult to reach from this relaxation. In this paper, we explore an alternative framework to generate cuts. We generate cutting planes that define facets of the convex hull of a discrete relaxation. More precisely, we use relaxed decision diagrams to compactly represent a set of integer points that contains all feasible solutions and potentially some infeasible solutions. Then, we find cutting planes that are facet-defining for the convex hull of these points.

Relaxed decision diagrams are appealing for this purpose as they may contain information about the discrete structure of the problem that is not captured by an LP relaxation. For example, relaxed decision diagrams can produce stronger bounds than LP relaxations for problems such as maximum independent set, maximum cut, and maximum 2-satisfiability (Bergman et al. 2013, 2016b). They also have the advantage that the strength of the relaxation is adjustable, which enables us to control the trade-off between the strength of the cuts and the time it takes to generate a cut. Lastly, these cuts are independent from the continuous relaxation that produces the fractional point to separate, and only require a (relaxed) decision diagram for the problem at hand.

In principle, the set of feasible solutions for every bounded pure integer program can be represented by decision diagrams (Behle 2007, Wegener 2000). That said, constructing small, yet strong, relaxed decision diagrams may be challenging, depending on the application. The approach of Bergman et al. (2013, 2016b) utilizes a dynamic programming formulation of the problem for

Preprint version

this purpose. Therefore, if strong problem-specific construction methods exist, we can take advantage of them in our generic method. For example, in our computational experiments we will use a construction method for the independent set problem that is known to produce relatively strong relaxations.

Decision diagrams have been previously applied to cut generation for integer programs by Becker et al. (2005); see also Behle (2007). In these works, the authors represent a subset of active constraints exactly with a decision diagram. Valid cuts are then generated via a Lagrangian relaxation, using the decision diagram as an optimization oracle. Our method has some similarities to these previous works, but it differs from them in several ways. From a formal standpoint, we investigate the correspondence between decision diagrams and polar sets, which enables us to prove that the generated cuts from exact decision diagrams are facet-defining. In practice, we use relaxed decision diagrams rather than exact decision diagrams for a subset of constraints. The strength of relaxed decision diagrams can be controlled by a maximum width parameter.

These cuts, directly generated from a polar set, are called target cuts, a term coined by Buchheim et al. (2008). The main difference between their approach and ours is that they use projection to make the separation problem tractable, while we use relaxed decision diagrams.

The embedding of decision diagram-based target cuts in integer programming solvers can be realized in different ways. The most natural integration is to build a (relaxed) decision diagram for the entire problem at hand. For several combinatorial optimization problems, such decision diagrams have already been developed and can be readily used (Bergman et al. 2016b). In principle, it is possible to automatically build decision diagrams for arbitrary integer programming models (Behle 2007), although the associated relaxed decision diagrams may not be sufficiently strong. Alternatively, it is possible to automatically generate a decision diagram for specific substructures of problem at hand, such as clique constraints, covering constraints, or partitioning constraints. The associated target cuts would then be facet-defining for those substructures, but also valid for the overall problem.

The remainder of the paper is structured as follows. In Section 2, we introduce the general concepts and notation regarding decision diagrams. We then review target cuts in Section 3. Section 4 provides the relationship between decision diagrams and polyhedra for binary integer programs. We introduce the target cut generation method in Section 5. Section 6 describes how to construct certificates of the dimension of faces defined by cuts. In Section 7, we generalize the results from binary integer programs to any bounded integer program. Section 8 contains extensive computational results on the independent set problem to assess the potential strength of the target cuts. Section 9 concludes this paper.

2 Decision Diagrams

A *binary decision diagram* (BDD) can be viewed as a graph representation of a set of binary points of dimension n, such as the feasible points of a binary IP with n variables. An example is shown in Figure 1. More precisely, a BDD is a directed acyclic multigraph that has the following layered structure. The nodes of a BDD are partitioned into $n+1$ layers, where the first n layers correspond to the n variables in some fixed order and the last layer is reserved for a single terminal node t. The first layer contains a single root node s. Arcs are similarly assigned to layers: the layer of an arc corresponds to the layer of its tail. Except for the terminal node which has no outgoing arcs, each node in layer k has at most two outgoing arcs incident to nodes in layer $k+1$. These arcs

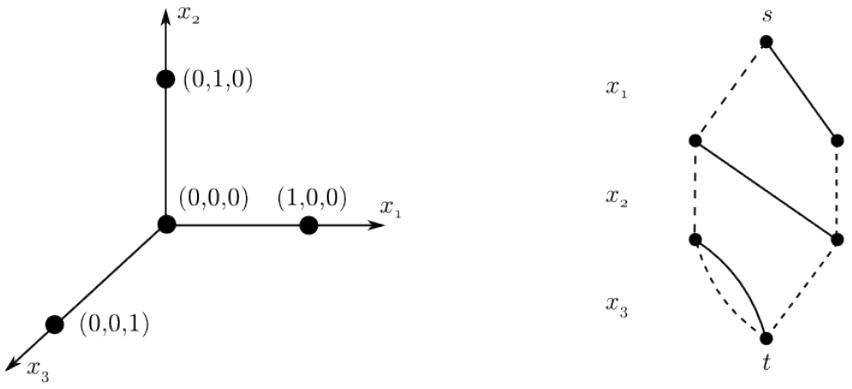

Figure 1: The BDD on the right represents the set of four points on the left. This BDD has a width of 2. A dashed line indicates a 0-arc and a full line indicates a 1-arc. Arcs are oriented from top to bottom. Note that each of the four paths from s to t in the BDD corresponds to a point in the set. For instance, following the rightmost arcs gives us the point $(1, 0, 0)$.

are labeled either 0 or 1 – called 0-arc and 1-arc respectively – and there can be at most one arc of each label leaving a node. Parallel arcs are allowed; that is, both 0-arc and 1-arc of a node may be incident to the same node. Moreover, every node except s and t has at least one incoming arc and at least one outgoing arc.

We can use the structure above to represent a set S of n-dimensional binary points through the following property: there is a one-to-one correspondence between each point $x \in S$ and each directed path from s to t in the BDD. Given a directed path p from s to t, the corresponding x is such that $x_k = 0$ if the layer-k arc of p is a 0-arc, or $x_k = 1$ if it is a 1-arc. In other words, a layer-k node has an outgoing ℓ-arc if and only if there exists $x \in S$ with $x_k = \ell$ given that $x_1, \ldots, x_{k-1}$ are similarly assigned according to the path from s to that node.

Note that a BDD is not unique for S. However, given a fixed variable ordering, a unique smallest BDD for S exists, called the *reduced* BDD (Knuth 2011). In this paper, we do not require the BDD to be reduced, but working with reduced BDDs inherently speeds up cut generation as its performance depends on the size of the BDD.

A BDD may be enhanced with *long arcs* in order to reduce the number of nodes. In the definition of BDDs above, arcs go from a layer to the next. A long arc is an arc that skips layers; that is, it may go from a layer to any higher layer. Long arcs in general may represent any set of assignments to the variables between the two layers, indicated by their labels. A common use of long arcs is to replace nodes that have a single outgoing 0-arc. In this form, a long 1-arc from layer k to $k + r$ represents the assignments $x_k = 1, x_{k+1} = 0, \ldots, x_{k+r} = 0$, while a long 0-arc represents $x_k = 0, \ldots, x_{k+r} = 0$. Such BDDs are called *zero-suppressed BDDs*, or ZDDs (Minato 1993). The cut generation and theoretical framework in this paper supports long arcs in its more general form. That is, it may represent any set of assignments $x_k = \ell_k, \ldots, x_{k+r} = \ell_{k+r}$, according to its label $(\ell_k, \ldots, \ell_{k+r})$.

The traditional application of BDDs in the context of Boolean functions uses two terminal

nodes, representing the 'true' and 'false' evaluation of the function, respectively. In our context it suffices to use only the 'true' terminal node to represent all points in S.

If the finite set of points to be represented is not binary, then a *multivalued decision diagram* (MDD) can be used. An MDD is an extension of a BDD that allows labels to be other than 0 and 1, and thus allows points of any value. In this work, we focus on BDDs first and then extend the results to MDDs.

Now that we have defined BDDs as a structure to represent a set of points, the next step is to view it in terms of a relaxation for a discrete optimization problem.

Given a binary optimization problem, it is typically not efficient to have a BDD represent its feasible set exactly, since it may have exponentially many nodes with respect to the size of the problem. As an approximation, we consider tractable BDDs that are only required to contain the feasible set and not represent it exactly. These BDDs are called *relaxed BDDs* (Bergman et al. 2011, 2016b) and are usually of size polynomial or linear in the size of the problem. Formally, given a set of points $S \subseteq \{0,1\}^n$, a relaxed BDD with respect to S is a BDD that represents a set of n-dimensional points S' that contains S. To differentiate relaxed BDDs from BDDs representing exactly the set of feasible points, we call the latter *exact BDDs*.

The general idea of constructing a relaxed BDD is to restrict the number of nodes of its largest layer, called *width*, during its construction in such a way that no feasible points are removed in the process. For details about the construction of a relaxed BDD in the context of certain binary optimization problems, see Bergman et al. (2016b). We do not focus on BDD construction in this paper: our cut generation method assumes that a relaxed BDD is already constructed.

A very useful property of BDDs is that, once built, they can be used to optimize any additively separable objective function – such as linear functions – over the set of points they represent. This can be done as follows: suppose that we want to maximize $\sum_{i=1}^{n} f_i(x_i)$. If we assign weights $f_i(\ell)$ to every ℓ-arc of layer i, then each maximum weight path of the BDD represents an optimal solution, due to the one-to-one correspondence between paths and feasible solutions. Given that a BDD is a directed acyclic graph, finding an optimal path can be done in time linear in the number of arcs of the BDD. Therefore, relaxed BDDs with a constant width yield dual bounds for the optimal value in linear time.

Binary decision diagrams were originally proposed in the context of circuit design and formal verification (Akers 1978, Bryant 1986, Lee 1959). Relaxed BDDs have a more recent history: they were initially introduced for constraint programming (Andersen et al. 2007) and further developed to obtain bounds for optimization problems through the above property (Bergman et al. 2011, 2016b, 2013). They have been shown to be effective for a number of practical applications such as disjunctive scheduling (Cire and van Hoeve 2013) and multi-dimensional bin packing (Kell and van Hoeve 2013).

3 Target Cuts

Before defining target cuts, we first recall a concept from convex analysis useful in cutting plane theory. The *polar set* of an n-dimensional set S is defined as

$$S^\circ = \{u \in \mathbb{R}^n : u^\top x \leq 1 \ \forall x \in S\}.$$

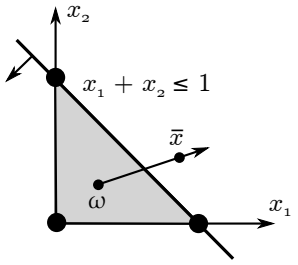

Figure 2: Geometrical interpretation of a target cut. A target cut defines a facet that intersects the segment between the point $\bar{x}$ to separate and an interior point ω of the polyhedron.

That is, the polar set can be viewed as the set of (coefficients of) all inequalities of the form $u^\top x \leq 1$ that are valid for S. If the origin happens to be in the interior of the convex hull of S, denoted by $\text{conv}(S)$, then the polar set corresponds to all inequalities valid for S. This is because any inequality strictly satisfied by the origin has a positive right-hand side and vice versa, and such inequality may be normalized to a right-hand side of 1.

An important property of polar sets is that each vertex of the polar set of S corresponds one-to-one to a facet of the convex hull of S, under the assumptions that $\text{conv}(S)$ is full-dimensional and the origin is in the interior of $\text{conv}(S)$. In other words, it allows us to relate the vertices of a polyhedron to its facets, and thus it is a natural channel through which we can generate valid inequalities from a discrete relaxation. For further details on polar sets, see e.g. Schrijver (1986).

Target cuts can be defined geometrically or algebraically (Buchheim et al. 2008). Their name comes from the geometric view illustrated in Figure 2 and they are defined as follows. The problem we consider is to separate some set S from a point $\bar{x} \notin \text{conv}(S)$. Let ω be an interior point of $\text{conv}(S)$ and assume $\text{conv}(S)$ is a polyhedron. If we shoot a ray from the interior point ω to the point $\bar{x}$ – that is, we view $\bar{x}$ as a "target to shoot at" – then a *target cut* is an inequality defined by a facet of $\text{conv}(S)$ intersected by the ray. Observe that the ray may intersect more than one facet by hitting a lower dimensional face, and thus more than one target cut may exist for given $\bar{x}$ and ω.

From an algebraic perspective, a target cut is a valid and facet-defining cut for $\text{conv}(S)$ and has maximum violation (for a normalized right-hand side of 1) with respect to $\bar{x}$ in the translated space where ω is the origin. Such an inequality is tightly connected to polar sets: its coefficients are given by an extreme point of $(S-\omega)^\circ$ with maximum violation. Therefore, we may formally define a target cut as an inequality $u^{*\top}(x-\omega) \leq 1$ where u^* is an optimal extreme point solution for the problem $\max\{u^\top(\bar{x}-\omega) : u \in (S-\omega)^\circ\}$. This inequality is valid and facet-defining for S since it is given by an extreme point of the polar set and it cuts off $\bar{x}$ since we maximize violation. The equivalence between the geometrical and algebraic definitions is given by Buchheim et al. (2008).

The choice of S is crucial: ideally we would like S to be the (integer) feasible set itself, but optimizing over its polar set is generally impractical as it has as many constraints as (integer) feasible solutions of the problem. Therefore, we let S be some tractable projection or relaxation of the feasible set of the problem, which still yields a cut valid for the feasible set. Note that since we want to cut off $\bar{x}$, this relaxation cannot be the one that yielded $\bar{x}$ or any strictly weaker relaxation.

Buchheim et al. (2008) make the separation problem for target cuts tractable by considering a sufficiently small projection of the polytope. Target cuts in this form have been computationally investigated for a number of applications: certain constrained quadratic 0-1 problems (Buchheim et al. 2010), the maximum cut problem (Bonato et al. 2014), the equicut problem (Anjos et al. 2013), and robust network design (Buchheim et al. 2011). Our method differs from their approach in that we turn the cut generation tractable by using relaxed decision diagrams instead of projection.

In other previous work, Boyd (1995) relates these cuts, under the name 1-polar cuts, to Fenchel cuts from a theoretical perspective. This method among others was also discussed by Cadoux and Lemaréchal (2013), under the assumption that an oracle for optimizing over the polar set is available. In addition, target cuts are closely related to local cuts (Chvátal et al. 2013), which support face certificates such as those described in Section 6.

4 Decision Diagrams and Polyhedra

The correspondence between paths and points in decision diagrams allows us to relate them to certain useful polyhedra. In this section, we first review a result from Behle (2007) connecting decision diagrams to the convex hull of the set of points they represent. Then, we show that polar sets can be obtained from decision diagrams in a similar fashion.

4.1 Convex hull

Let S be a set of binary points of dimension n, for example the feasible set of a binary IP. Denote by $\text{proj}_x(P)$ the projection of P onto x. That is, $\text{proj}_x(P) := \{x : \exists y \text{ s.t. } (x,y) \in P\}$. Let us first write down the definition of convex hull:

$$\text{conv}(S) = \left\{ x \in \mathbb{R}^n : \exists \alpha \text{ s.t. } \sum_{\hat{x} \in S} \alpha_{\hat{x}} \hat{x} = x,\ \sum_{\hat{x} \in S} \alpha_{\hat{x}} = 1,\ \alpha \geq 0 \right\}.$$

Here, α takes the role of the coefficients through which x can be expressed as a convex combination of the points of S. In the context of BDDs, the role of α can be taken by flow variables as in the theorem below. Denote by $\delta^+(i)$ and $\delta^-(i)$ respectively the set of outgoing arcs from i and incoming arcs to i.

Theorem 1 (Behle 2007)**.** *Consider a BDD B with vertices V and arcs A representing the set of points S, with root s and terminal t. Assume B has no long arcs. Let S_k be the set of 1-arcs at layer k. Define*

$$\begin{aligned} P_{flow}(B) = \{(f,x) \in \mathbb{R}^{|A|} \times \mathbb{R}^n \ : \ & \sum_{j \in \delta^-(i)} f_{ji} - \sum_{j \in \delta^+(i)} f_{ij} = 0 && \forall i \in V \setminus \{s,t\}, \\ & \sum_{j \in \delta^+(s)} f_{sj} = 1, \\ & \sum_{(i,j) \in S_k} f_{ij} = x_k && \forall \text{ layer } k, \\ & f_{ij} \geq 0 && \forall (i,j) \in A\}. \end{aligned}$$

Then

$$proj_x(P_{flow}(B)) = conv(S).$$

Compare the above formulation with the original convex hull formulation. The first two classes of constraints in $P_{\text{flow}}(B)$ indicate that f is a feasible flow of value one on the BDD B, which take the role of ensuring that the coefficients α of the convex combination sum up to one. The third constraint ensures that the total flow through the 1-arcs of each layer k is equal to x_k, which mirrors expressing x in terms of the coefficients α.

More precisely, a feasible flow of value 1 can be viewed as an edge weight vector representing the weights of a convex combination of paths from s to t. Since each path from s to t in the BDD corresponds to a point of S and vice versa, the flow can be translated to the coefficients of a convex combination of the points of S that yields x.

Note that the above theorem not only gives us a way to check if a point x is in $\text{conv}(S)$ or not, but it also allows us to explicitly express x as a convex combination of the points of S. This can be done by decomposing the flow into weighted paths.

If the BDD has long arcs, the theorem can be adapted by generalizing S_k. Let S_k instead be the set of arcs that have $x_k = 1$ as part of their labels. The rest of the theorem remains the same.

More generally, a similar characterization can also be obtained for dynamic programming formulations (Martin et al. 1990).

4.2 Polar set

We now relate a BDD representing a binary set of points S to the polar set of S, defined by $S^\circ = \{u \in \mathbb{R}^n : u^\top x \leq 1 \ \forall x \in S\}$. As previously discussed, polar sets are essential to the generation of target cuts.

Consider again a BDD B with vertices V, arcs A, root s, and terminal t representing S. Assume B has no long arcs for now. We define a polyhedron $P^*(B)$ and show that it formulates S° in a higher dimension.

Our model is constructed with the following intuition. First, view the constraints of the polar set as $1 - \sum_{k=1}^n u_k x_k \geq 0$ for each $x \in S$. That is, if we start with a budget of 1 and pay a cost of u_k whenever $x_k = 1$, we must end up with a nonnegative value.

We translate this intuition into the context of decision diagrams. Define variables v_i for each node i and u_k for each layer k. Start at s with a budget of $v_s = 1$ and traverse the BDD towards t. For each layer-k 1-arc traversed, we pay a cost of at least u_k. The value v_i at each node i must be consistent with every path leading to i. Finally, when we reach t, we must have at least $v_t = 0$ remaining for every path up to t. Note that u and v may be negative.

This intuition leads to the following formulation.

$$\begin{aligned} P^*(B) = \{(u,v) \in \mathbb{R}^n \times \mathbb{R}^{|V|} : \ & v_j \leq v_i - u_k && \forall \text{ 1-arc } (i,j) \text{ of layer } k, \\ & v_j \leq v_i && \forall \text{ 0-arc } (i,j), \\ & v_s = 1, \\ & v_t = 0\} \end{aligned}$$

In other words, for each path corresponding to x we pay a total of $\sum_{k=1}^n u_k x_k$ from a budget of 1 to traverse from s to t, so we ensure that $u^\top x \leq 1$ for all $x \in S$. We formalize this interpretation below.

Theorem 2. $\text{proj}_u(P^*(B)) = S^\circ$.

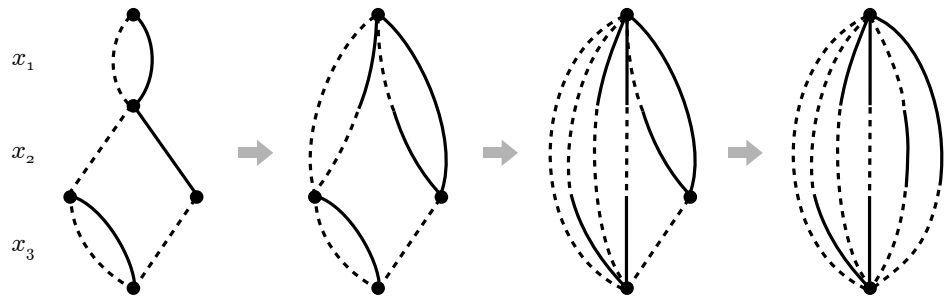

Figure 3: Process described in the alternative proof of Theorem 2: projecting out of the node variables iteratively until we reach an explicit description of the represented set. Dashed and full parts of long arcs indicate the labels of 0 and 1 respectively for the corresponding layer.

Proof. ($\subseteq$) Let $u \in \text{proj}_u(P^*(B))$. We want to show that $u^\top x \leq 1$ for all $x \in S$. Let $x \in S$. Then x corresponds to a path p from s to t in the BDD. For each layer-k arc (i,j) in p, we have $v_j \leq v_i$ if $x_k = 0$ or $v_j \leq v_i - u_k$ if $x_k = 1$. By summing up all these constraints, they imply $v_t \leq v_s - u^\top x$. Since $v_s = 1$ and $v_t = 0$, $u^\top x \leq 1$. Hence $u \in S^\circ$.

($\supseteq$) Let $u \in S^\circ$, that is, $u^\top x \leq 1$ for all $x \in S$. We want to show that there exists v such that the constraints of $P^*(B)$ are satisfied. Define

$$v_s = 1,\ v_t = 0$$
$$v_j = \min \left\{ \min_{\text{1-arc } (i,j) \text{ of layer } k} \{v_i - u_k\}, \min_{\text{0-arc } (i,j)} \{v_i\} \right\} \quad \forall j \in V \setminus \{s,t\}.$$

By construction of v, all constraints are satisfied except possibly the constraints for layer-n arcs (i,t). To show they are also satisfied, define $\hat{v}_t := \min \left\{ \min_{\text{1-arc } (i,t)} \{v_i - u_n\}, \min_{\text{0-arc } (i,t)} \{v_i\} \right\}$, which is the definition of v_j above applied to $j = t$. It suffices to show that $v_t \leq \hat{v}_t$ since this encapsulates all arc constraints for layer n.

The definitions of v and $\hat{v}_t$ imply that there is a path p from s to t such that $\hat{v}_t = v_s - \sum_{k \in S_p} u_k$, where S_p is the set of layers in which p has a 1-arc. In other words, there exists an $x \in S$ such that $\hat{v}_t = 1 - u^\top x$. Since $u \in S^\circ$, we have $1 - u^\top x \geq 0$, and hence $\hat{v}_t \geq 0 = v_t$.

Therefore, all constraints of $P^*(B)$ are satisfied for (u, v) and thus $u \in \text{proj}_u(P^*(B))$. □

Theorem 2 assumes that B has no long arcs, but it can be adapted to consider them in their general form. To do so, it suffices to generalize the arc constraints in $P^*(B)$. Suppose that we have a long arc (i,j) from layer k to $k+r$ labeled with the partial solution $x_k = \ell_k, \ldots, x_{k+r} = \ell_{k+r}$. Then we write its corresponding constraint in $P^*(B)$ as $v_j \leq v_i - \sum_{p=k}^{k+r} \ell_p u_p$. The first part of the above proof still holds for this variant: by summing up the arc constraints in a path representing x including the long arcs, we still obtain $u^\top x \leq 1$. The second part also holds if we generalize v_j to $\min_{\text{arc } (i,j) \text{ of label } \ell} \{v_i - \sum_{p=k}^{k+r} \ell_p u_p\}$.

In light of the idea of long arcs, we present an alternative proof for Theorem 2 that provides additional insight.

Alternative proof for Theorem 2. We consider what occurs when we apply one step of Fourier-Motzkin elimination in order to project out a variable v. Here we assume long arcs may exist.

In our context, applying Fourier-Motzkin elimination to v is reduced to summing up the constraints of each pair of incoming and outgoing arcs of v. More specifically, for each pair of incoming arc (i, v) labeled ℓ^{in} and outgoing arc (v, j) labeled ℓ^{out}, we replace their corresponding constraints by the constraint $v_j \leq v_i - \sum_{p \in R} \ell_p u_p$, where ℓ is the label ℓ^{in} concatenated with ℓ^{out} and R is the range between the layers of i and j. We may view this as replacing these arcs by a long arc with label ℓ.

Therefore, if we apply Fourier-Motzkin elimination to all nodes of the BDD except s and t, the result is a graph with one long arc for each original path with the label of the corresponding point x, as illustrated in Figure 3. This implies $\text{proj}_u(P^*(B)) = \{u \in \mathbb{R}^n : 0 \leq 1 - u^\top x \; \forall x \in S\} = S^\circ$. □

This proof shows us that we can replace nodes of the BDD by long arcs until we reach an explicit description of all points through long arcs, and in practice any intermediate BDD B constructed in this process can be used to formulate $P^*(B)$. For each node v we project out, we replace $\text{in}(v) + \text{out}(v)$ arcs by $\text{in}(v) \cdot \text{out}(v)$ arcs, where $\text{in}(v)$ and $\text{out}(v)$ is the number of incoming and outgoing arcs of v respectively. In particular, we can remove any node with a single incoming arc or a single outgoing arc and reduce the number of arcs by one. We can also remove any node with two incoming arcs and two outgoing arcs without increasing the number of arcs. This process allows us to construct a more compact version of $P^*(B)$.

As previously discussed, generating target cuts requires us to work with the polar of a translated set S. We adapt Theorem 2 to characterize $(S - \omega)^\circ$ with BDDs for any point ω. Doing so only requires a modification of the $v_s = 1$ constraint in $P^*(B)$. Define

$$\begin{aligned} P^*_\omega(B) = \{(u, v) \in \mathbb{R}^n \times \mathbb{R}^{|V|} : \; & v_j \leq v_i - u_k && \forall \text{ 1-arc } (i, j) \text{ of layer } k, \\ & v_j \leq v_i && \forall \text{ 0-arc } (i, j), \\ & v_s = 1 + u^\top \omega, \\ & v_t = 0\}. \end{aligned}$$

Then, given S as the set of points represented by B,

Theorem 3. $\text{proj}_u(P^*_\omega(B)) = (S - \omega)^\circ$.

Proof. The proof of Theorem 2 can be adapted to prove this theorem. Note that $(S - \omega)^\circ = \{u : u^\top (x - \omega) \leq 1 \; \forall x \in S\} = \{u : u^\top x \leq 1 + u^\top \omega \; \forall x \in S\}$. In the proof of Theorem 2, v_s leads to the right-hand side of the constraints in the polar set. Therefore, instead of replacing v_s by 1, replace v_s by $1 + u^\top \omega$, and the same steps lead to this result. □

Theorem 3 is not only useful for our cut generator in this paper, but it also provides theoretical insight on the representability of BDDs with respect to polyhedra. Define a *sub-BDD* of a BDD B as a subgraph of B structured as a BDD with the same root and terminal nodes.

Corollary 1. *Let B be a BDD representing S. The set of binary points of any face of conv(S) is representable by a sub-BDD of B.*

Proof. Let F be a face of $\text{conv}(S)$. Let ω be a point in the relative interior of $\text{conv}(S)$. Denote by S_ω the translation of S so that it contains the origin in its relative interior, that is, $S_\omega = S - \omega$. Additionally, consider the similarly translated $F_\omega = F - \omega$.

Consider a valid inequality $u^\top x \leq v$ defining F_ω. That is, $\{x \in \text{conv}(S_\omega) : u^\top x = v\} = F_\omega$. Since the origin is satisfied by the inequality, $v \geq 0$. Without loss of generality, we consider the cases $v = 0$ and $v = 1$, since the inequality can be normalized when $v > 0$.

Suppose the inequality is of the form $u^\top x \leq 0$. Then it must be an implicit equality – in other words, $u^\top x = 0$ is valid – because otherwise the origin would not be in the relative interior of $\text{conv}(S_\omega)$. Then $F_\omega = S_\omega$, and thus $F = S$, which is representable by B.

Suppose the inequality is of the form $u^\top x \leq 1$. This implies $u \in S_\omega^\circ$. By Theorem 3, there exists v such that $(u, v) \in P_\omega^*(B)$. We call an arc tight if its corresponding constraint for $P_\omega^*(B)$ is tight for (u, v). That is, an ℓ-arc (i, j) at layer k is tight if $v_j = v_i - \ell u_k$ (or, if (i, j) is a long arc, $v_j = v_i - \sum_p \ell_p u_p$). We call a path tight if it is composed only of tight arcs. Let T be the set of arcs that are in at least one tight path from the root s to the terminal t. The subgraph of B induced by T forms a sub-BDD B'.

Let S' be the set of points represented by B'. Then $x \in S'$ if and only if x corresponds to a tight path from s to t, or equivalently $u^\top x = 1 + u^\top \omega$ by summing up all constraints of the path. This implies $S' = \{x \in \text{conv}(S) : u^\top (x - \omega) = 1\} = F_\omega + \omega = F$. Therefore B' represents the set of binary points of the face F. □

In other words, if we take the sub-BDD given by the paths corresponding to the points of a face, we do not end up including a path corresponding to a point not in the face.

Corollary 1 establishes that it is easy to represent the points of any face of a polytope corresponding to a BDD if it is easy to represent the points of the BDD itself. This can also be interpreted as a negative result: if a face of $\text{conv}(S)$ is represented by a reduced BDD with N nodes (recall that reduced BDDs are unique), then we cannot hope to construct an exact BDD representing S with less than N nodes with the same variable ordering.

As a final remark to this section, there is a good reason why Theorems 1 and 2 are similar: they are connected by duality. Given $\lambda \geq 0$, define $P_\lambda(B)$ as $P_{\text{flow}}(B)$ except that we replace the constraint $\sum_{j \in \delta^+(s)} f_{sj} = 1$ by $\sum_{j \in \delta^+(s)} f_{sj} = \lambda$. It is possible to slightly adapt Theorem 1 so that we have a more general result: $\text{proj}_x(P_\lambda(B)) = \lambda \, \text{conv}(S)$.

Then, given x and assuming solutions to the problems below exist, we have the following duality relationship:

$$\min\{\lambda \in \mathbb{R}_+ : \exists f \text{ s.t. } (f, x) \in P_\lambda(B)\} = \max\{u^\top x : \exists v \text{ s.t. } (u, v) \in P^*(B)\}.$$

This mirrors a duality relationship known in convex analysis: given a closed convex set C containing the origin, the gauge function of C is equal to the support function of C° – see e.g. Hiriart-Urruty and Lemaréchal (2001). The gauge function of a closed convex set C containing the origin is defined as $\inf\{\lambda > 0 : x \in \lambda C\}$ and the support function of a nonempty set C is defined as $\sup\{\langle s, x\rangle : s \in C\}$. In our context, this applies to $C = \text{conv}(S)$.

5 Target Cut Generation from Decision Diagrams

5.1 Cut generation algorithm

The theory developed in the previous section is useful to generate target cuts with respect to decision diagrams. Our cut generating algorithm consists of two steps:

1. construct a relaxed decision diagram B, and
2. optimize over $P^*_\omega(B)$ to generate a cut.

In this work, we assume that a relaxed decision diagram is already constructed for us and we now show that the second step yields a valid cut.

We first consider the separation problem with respect to an arbitrary polyhedron P: given a point $\bar{x}$, we would like to either find an inequality valid for P and violated by $\bar{x}$, or decide that $\bar{x} \in P$. Ideally, we would like P to be the integer hull of the problem, but that is typically intractable. We return to the choice of P after discussing how to obtain a valid cut.

The following classic result in polarity theory, discussed for instance in Schrijver (1986), captures an important property of polar sets.

Theorem 4. *Let P be a full-dimensional polyhedron containing the origin as an interior point. Then $u^\top x \leq 1$ is a facet-defining inequality of P if and only if u is an extreme point of P°.*

Moreover, given a finite set S, $\text{conv}(S)^\circ = S^\circ$ when $\text{conv}(S)$ contains the origin. This allows us to work directly with S° to generate facet-defining inequalities for $\text{conv}(S)$.

The condition on the origin can be replaced if we are given an interior point ω of P. In this case, we may translate P so that ω is in the origin and apply the above theorem. Note that we must also translate the resulting inequality back, as detailed below.

Corollary 2. *Let P be a full-dimensional polyhedron and let ω be an interior point of P. Then $u^\top(x - \omega) \leq 1$ is a facet-defining inequality of P if and only if u is an extreme point of $(P - \omega)^\circ$.*

Finally, we may view the above corollary from the perspective of cut generation as follows.

Proposition 1. *Let P be a full-dimensional polyhedron. Let ω be an interior point of P, a point $\bar{x}$ we want to separate, and*

$$u^* = \arg\max_u \{u^\top(\bar{x} - \omega) : u \text{ is an extreme point of } (P - \omega)^\circ\}.$$

Then $u^{*\top} x \leq 1 + u^{*\top}\omega$ is a facet-defining cut for $\bar{x}$ if $u^{*\top}(\bar{x} - \omega) > 1$, or $\bar{x} \in P$ otherwise.

Proof. The optimal point u^* can be attained because the polar of a set containing the origin in its interior is bounded. By Corollary 2, the inequality is facet-defining for P. Moreover, the inequality is a valid cut for $\bar{x}$ because we maximize the violation of the inequality with respect to $\bar{x}$, and thus a cut will be found if and only if $\bar{x} \notin P$. □

Proposition 1 summarizes target cut generation. We return to the choice of P. We would like P to be a tractable polyhedron that is close to the integer hull of the problem and yields valid cuts, such as a relaxation of the integer hull. Buchheim et al. (2008) chooses P to be a sufficiently small projection of the integer hull.

In our algorithm, we let P be the convex hull of the set S of points represented by a relaxed decision diagram B. Any inequality valid for a relaxation of the problem is also valid for the problem itself, and therefore we may use this method to generate valid cuts if we can optimize over S°. We have shown in the previous section that $P^*(B)$ formulates S° in a higher dimensional space, and thus if the decision diagram has a small enough number of arcs, then generating a target cut should be tractable.

Therefore, the algorithm can be concisely summarized as follows. Given a relaxed decision diagram B representing S, optimize over $P^*_\omega(B)$ for some interior point ω of $\mathrm{conv}(S)$, and the resulting point is a valid cut for $\bar{x}$ if $\bar{x} \notin \mathrm{conv}(S)$. A more precise description is given in Algorithm 1, which also includes details addressed later in this section.

So far we have assumed that $\mathrm{conv}(S)$ is full-dimensional and an interior point ω is known. We eliminate these assumptions in the following subsections. Finally, at the end of this section we discuss the degenerate case where an extreme point of $P^*(B)$ does not project down to an extreme point of $\mathrm{conv}(S)$, required for the facet-defining property of the cut.

Algorithm 1 BDD-based target cut generation algorithm

Input: Relaxed (or exact) BDD B representing a set of points S and point $\bar{x}$ to separate.
Output: Decide whether $\bar{x} \in \mathrm{conv}(S)$ or not. If $\bar{x} \notin \mathrm{conv}(S)$, return a valid inequality or equality that separates $\bar{x}$ from $\mathrm{conv}(S)$.

BDD-Target-Cut-Generator$(B, \bar{x})$
$\omega \leftarrow$ relative interior point of $\mathrm{conv}(S)$ (e.g. the geometric center of S)
find an optimal solution or unbounded ray (u^*, v^*) to the following problem:

$$\begin{aligned} \max\ & u^\top(\bar{x} - \omega) \\ & v_j \le v_i - u_k && \forall \text{ 1-arc } (i,j) \text{ of layer } k, \\ & v_j \le v_i && \forall \text{ 0-arc } (i,j), \\ & v_s = 1 + u^\top \omega, \\ & v_t = 0. \end{aligned} \tag{P}$$

if (u^*, v^*) is an optimal solution **then**
  if $u^{*\top}(\bar{x} - \omega) \le 1$ **then**
    return $\bar{x} \in \mathrm{conv}(S)$ ▷ i.e., a cut cannot be derived from the relaxation B
  else
    optionally ensure u^* is an extreme point of S° (see Section 5.4)
    return $u^{*\top} x \le 1 + u^{*\top}\omega$, a cut separating $\bar{x}$ from $\mathrm{conv}(S)$
else if (u^*, v^*) is an unbounded ray **then**
  return $u^{*\top} x = u^{*\top}\omega$, a valid equality separating $\bar{x}$ from $\mathrm{conv}(S)$

5.2 Non-full-dimensional case

We have assumed so far that $\mathrm{conv}(S)$ is full-dimensional. Two modifications to the algorithm are required to remove this assumption: the point ω is generalized to be in the relative interior of

conv(S) rather than the interior, and we must handle the case when the problem is unbounded.

If the problem is bounded, the algorithm remains the same. If the problem is unbounded, then let r^* be an unbounded ray. Buchheim et al. (2008) show that ${r^*}^\top(x - \omega) = 0$ is a valid equality that is violated by $\bar{x}$. Thus the algorithm may return this equality.

5.3 Obtaining an interior point

A relative interior point ω can be obtained from B. The geometric center of S – that is, the arithmetic mean of all points of S – can be obtained in time linear in the number of nodes of B (Behle 2007), which can be used for this purpose. Note that the center of S is in the relative interior of conv(S) because it can be expressed as a convex combination of all vertices of conv(S) with (positive) uniform coefficients. We describe in Appendix A an alternative algorithm to obtain the geometric center of S that is more suitable to our context.

Boyd (1995) argues that one must be careful when selecting the interior point because the resulting facet depends on this choice. For example, if the point is too close to a facet, then from the geometrical interpretation of target cuts we can see that the algorithm prioritizes that facet. With this intuition in mind however, the center of S appears to be a balanced choice of interior point.

This approach may not be necessary for certain problems. For example, it is easy to find an interior point for the independent set problem, as discussed in Section 8.1.

5.4 Ensuring a facet-defining cut

There is one final detail to complete the algorithm. The cut generator as described so far will produce a cut for $\bar{x}$, but it may not be facet-defining with respect to the convex hull of the points S defined by the BDD relaxation.

An extreme point of the polar set is required in order to obtain a facet-defining cut. If we use the simplex algorithm, we obtain an extreme point (u^*, v^*) of $P^*_\omega(B)$. However, u^* is not necessarily an extreme point of $\text{proj}_u(P^*_\omega(B)) = (S - \omega)^\circ$. Note that this may only happen in the degenerate case where u^* is not a unique optimal solution.

We propose two methods to obtain an extreme point of $(S-\omega)^\circ$: an exact method and a heuristic method. The former requires n invocations of an LP solver, which may make it impractical for larger instances, but it is theoretically relevant as it produces a facet-defining cut in polynomial time.

Assume that the solution is not an extreme point of $\text{proj}_u(P^*_\omega(B))$. Let V^* be the optimal value of the problem, that is, $V^* = {u^*}^\top(\bar{x} - \omega)$. First note that u^* lies on the face F of $\text{proj}_u(P^*_\omega(B))$, where $F = \{u \in \text{proj}_u(P^*_\omega(B)) : u^\top(\bar{x} - \omega) = V^*\}$. We are interested in finding an extreme point of F, as it is also an extreme point of $\text{proj}_u(P^*_\omega(B))$ and it is optimal.

The following algorithm ensures that the solution is an extreme point of F. First, add the constraint $u^\top(\bar{x} - \omega) = V^*$ to the LP. For $j = 1, \ldots, n$, iteratively reoptimize the problem by maximizing u_j, and at the end of each iteration add the constraint $u_j = v^j$, where v^j is the optimal value at iteration j. At the end of this algorithm, we obtain an extreme point of F. To see why, let F^j be F with the addition of these constraints up to iteration j. Throughout the algorithm, we maintain the property that F^j is a face of F. Since F^n must be a single point and it is a face of F, the final solution is an extreme point of F. Note that we can stop any time none of the reduced

costs of the nonbasic variables are zero, since then the solution is unique and hence an extreme point of F.

The systematic method above may be too slow for efficient application in practice. A faster heuristic is as follows. As before, first add the constraint $u^\top(\bar{x} - \omega) = V^*$ to the LP. Then apply a small random perturbation to the objective with respect to the u variables and reoptimize. The resulting solution is likely to be unique since, for any polytope, optimizing over a uniform random direction yields a unique solution with high probability.

6 Face Certificates

Duality allows us to numerically certify that a target cut separating $\bar{x}$ from S is facet-defining for $\text{conv}(S)$ in the case S is given explicitly. If S is given implicitly in a decision diagram, we may not obtain a facet-defining cut directly from the cut generating LP as discussed in the previous section. Nevertheless, we can still find a lower bound for the dimension of the face defined by the cut and certify it.

Define a *k-face certificate* for an inequality $u^\top x \leq 1$ as a set C of k affinely independent points in S such that $u^\top x = 1$ for all $x \in C$. It certifies by definition that the face given by $\{x \in \text{conv}(S) : u^\top x = 1\}$ has dimension at least k.

Our goal is to find face certificates when S is implicitly represented as a decision diagram. Recall that the LP we solve to generate a target cut from a decision diagram B with nodes V and arcs A is $\max_{u,v}\{u^\top \bar{x} : (u, v) \in P^*(B)\}$, denoted by (P). Its dual is $\min_f\{\sum_{j\in\delta^+(s)} f_{sj} : f \in \tilde{P}_{\text{flow}}\}$, denoted by (D), where $\tilde{P}_{\text{flow}} = \{f \in \mathbb{R}^{|A|} : \sum_{j\in\delta^-(i)} f_{ji} - \sum_{j\in\delta^+(i)} f_{ij} = 0 \ \forall i \in V \setminus \{s, t\},\ \sum_{(i,j)\in S_k} f_{ij} = \bar{x}_k \ \forall \text{ layer } k,\ f \geq 0\}$.

In order to obtain face certificates, we apply flow decomposition to an optimal value of (D). Consider the following class of flow decomposition algorithms, which turns a flow f into weights α for all paths (points in S). The algorithm below, which we call *standard flow decomposition*, iteratively selects paths and absorbs all possible flow in each step.

1. Choose a path p with positive flow according to any criteria. Let e be the bottleneck arc of p: $e = \arg\min_{e'\in p} f_{e'}$. Set α_x to f_e, where x is the point corresponding to p.
2. Reduce the flow of the path p by f_e. That is, set $f_{e'}$ to $f_{e'} - f_e$ for all $e' \in p$.
3. Repeat steps 1 and 2 until the value of the flow becomes zero.
4. Set all remaining α_x to zero.

The following theorem assures that, after solving (P) to optimality, if we apply a standard flow decomposition to its dual (D) and obtain k points with positive flow, then the dimension of the face of $\text{conv}(S)$ defined by the cut is at least k. Denote by S^+_α the set $\{x \in S : \alpha_x > 0\}$ for a given α.

Theorem 5. *Let (u^*, v^*) and f^* be an optimal primal-dual pair for (P) and (D). If f^* is an extreme point of $\tilde{P}_{flow}$ and α^* is the result of a standard flow decomposition applied to f^*, then $S^+_{\alpha^*}$ is a $|S^+_{\alpha^*}|$-face certificate for $u^{*\top}x \leq 1$ with respect to conv(S).*

Proof. See Appendix B. □

In the remainder of this section, we consider the case where $\text{conv}(S)$ is used as a relaxation of the problem and we would like to find face certificates with respect to the integer hull P_I of the problem rather than $\text{conv}(S)$. In other words, we seek a flow decomposition algorithm that finds a set S^+_α with many points in P_I.

First note that finding S^+_α that maximizes $|P_I \cap S^+_\alpha|$ is NP-hard. If we applied such a method to any flow that is positive in all arcs, it would allow us to identify whether a decision diagram contains a point in P_I or not. This is NP-hard because it generalizes the 0-1 IP feasibility problem, if we consider the width-1 decision diagram representing $\{0,1\}^n$.

Instead, we use a simple heuristic criterion: at each step of the decomposition, we choose a path that minimizes the sum of violations across all constraints of the corresponding IP. Finding this path is a matter of optimizing over the BDD, which can be done efficiently.

These certificates have further practical uses other than being a measure of strength. For instance, since they can be computed relatively quickly, they may be used in a rule to determine whether to continue generating more cuts or not. Moreover, any point from a certificate for P_I is a feasible point for the problem and may be used as a primal bound.

7 Multivalued Decision Diagrams

Up to the previous section, we have only considered binary decision diagrams. In this section, we extend the theoretical framework and cut generator to multivalued decision diagrams (MDDs). A multivalued decision diagram is a generalization of a binary decision diagram where we allow arcs to represent any value from a finite set, not only 0 or 1. This allows us to represent the feasible set of any bounded pure integer program.

Theorems 1, 2, and 3 may be adapted to the case the decision diagram is multivalued. Let M be an MDD with vertex set V, arc set A, root node s, and terminal t. Let S be the set of points M represents.

We start with Theorem 1. Denote by D_k the (finite) domain of variable k; that is, x_k may take any value in D_k. For every layer k and $\ell \in D_k$, let $S_{k,\ell}$ be the set of arcs that have $x_k = \ell$ as part of their labels. We generalize P_{flow} by adapting only the third class of constraints, as shown in the following formulation:

$$\begin{aligned} P_{\text{flow,MDD}}(M) = \{(f,x) \in \mathbb{R}^{|A|} \times \mathbb{R}^n \ : \ & \sum_{j \in \delta^-(i)} f_{ji} - \sum_{j \in \delta^+(i)} f_{ij} = 0 && \forall i \in V \setminus \{s,t\}, \\ & \sum_{j \in \delta^+(s)} f_{sj} = 1, \\ & \sum_{\ell \in D_k} \sum_{(i,j) \in S_{k,\ell}} \ell f_{ij} = x_k && \forall \text{ layer } k, \\ & f_{ij} \geq 0 && \forall (i,j) \in A\}. \end{aligned}$$

Then

Theorem 6. $\text{proj}_x(P_{flow,MDD}(M)) = conv(S)$.

Theorem 6 holds for the same reason as Theorem 1: a flow f in $P_{\text{flow,MDD}}(M)$ may be viewed as a convex combination of paths, and thus x corresponds to a convex combination of points in S.

This modification only generalizes the correspondence between f and x to accept values other than 0 and 1.

We now adapt Theorem 2. Assume M has no long arcs for now. We generalize the polytope P^* from Section 4.2 as follows:

$$P^*_{\text{MDD}}(M) = \{(u,v) \in \mathbb{R}^n \times \mathbb{R}^{|V|} : v_j \leq v_i - \ell u_k \qquad \forall\ \ell\text{-arc } (i,j) \text{ of layer } k,$$
$$v_s = 1,\ v_t = 0\}.$$

If M has long arcs, then the same modification from Section 4.2 applies here: if the label of the long arc is $(\ell_k, \ldots, \ell_{k+r})$, then we generalize the term ℓu_k above to $\sum_{p=k}^{k+r} \ell_p u_p$.

Similarly to Theorem 2, this polytope can be projected to the space of the u variables to obtain the polar set of S.

Theorem 7. $\text{proj}_u(P^*_{MDD}) = S^\circ$.

The alternative proof of Theorem 2 from Section 4.2 already takes into account general labels, and thus it proves Theorem 7 without any changes.

Finally, let $P^*_{\omega,\text{MDD}}(M)$ be equal to $P^*_{\text{MDD}}(M)$ except that we replace the constraint $v_s = 1$ by $v_s = 1 + u^\top \omega$. The theorem below generalizes Theorem 3 for MDDs.

Theorem 8. $\text{proj}_u(P^*_{\omega,MDD}) = (S - \omega)^\circ$.

Theorem 8 can be shown through the same line of reasoning as in Theorem 3.

Given that the entire theoretical framework from Section 4 can be generalized to MDDs, the cut generation algorithm also follows. In order to generate a target cut from a relaxed MDD M, we apply Algorithm 1, except that we optimize over $P^*_{\omega,\text{MDD}}(M)$ rather than $P^*_\omega(B)$ for a BDD B.

The theorems above show that MDDs can be used to generate target cuts for general bounded IPs. However, for a given combinatorial structure, a binary formulation may work best for an IP solver, but at the same time an MDD may be more natural or compact than a corresponding BDD. We next show how we can use an MDD to generate cuts for the binary IP formulation. More precisely, for each variable x_k with value ℓ in the MDD, we define variables y_{kp} with value 1 when $p = \ell$ or 0 otherwise, for $j = 1, \ldots, L$. These variables will be used to represent the binary IP model.

Let S_{bin} be the feasible set of this binarized IP. Then we may directly obtain the polar set of S_{bin}, allowing us to generate cuts for a binary IP from an MDD. Define

$$P^*_{\text{MDD,bin}}(M) = \{(u,v) \in \mathbb{R}^{nL} \times \mathbb{R}^{|V|} : v_j \leq v_i - u_{k\ell} \qquad \forall\ \ell\text{-arc } (i,j) \text{ of layer } k,$$
$$v_s = 1,\ v_t = 0\}.$$

Corollary 3. $\text{proj}_u(P^*_{MDD,bin}) = S^\circ_{bin}$.

Proof. We utilize the equivalence of the MDD with an associated BDD. Suppose that we replace a variable x_k by the corresponding binary variables y_{kp} for $p = 1, \ldots, L$. This transformation can be viewed in the MDD as replacing each ℓ-arc by a long arc with label $y_{k\ell} = 1$ and $y_{kp} = 0$ for $p \neq \ell$. The resulting decision diagram is a BDD B with long arcs. Observe that $P^*(B)$ is exactly the polyhedron $P^*_{\text{MDD,bin}}(M)$ above. Note that since we only replaced the arc labels, the graph structures of the MDD and BDD are the same. □

Therefore, we may optimize over $P^*_{\text{MDD,bin}}(M)$ to generate cuts from an MDD M for the corresponding binary IP.

8 Computational Results

We computationally investigate target cuts from relaxed decision diagrams from two points of view: strength and practicality. To evaluate the strength of the cuts, we observe the objective gap closed by the cuts and the dimensions of the faces they define. We do so on smaller instances for which we can construct the exact BDD, as we are interested in how they vary with the strength of the relaxation. Next, we examine overall solving times and size of the branch-and-bound tree on larger instances to determine how practical these cuts are. This takes into consideration the time it takes to generate them. Finally, we compare them to the Lagrangian approach proposed by Becker et al. (2005). Implementation and instances are available in `https://www.github.com/ctjandra/ddopt-cut`.

Before presenting the computational results, we first describe the instances we select and further details of our experimental setup.

8.1 Instance selection

We test our cut generation algorithm on two problem classes: the maximum independent set problem and the minimum set covering problem, both unweighted. Our goal is not necessarily to improve upon the state of the art of solving these problems, but to evaluate the advantages and disadvantages of target cuts from relaxed decision diagrams.

In particular, we require strong relaxations to generate cuts that can improve the performance of a MIP solver. Thus, in order to extract meaningful observations about these cuts, we choose instances of these problem classes for which sufficiently strong relaxations exist. These are high density graphs for independent set and low bandwidth instances for set covering, which we detail in the following subsections.

8.1.1 Maximum independent set problem

The maximum independent set problem is to find the maximum number of pairwise nonadjacent vertices in a given graph. Previous work provides an efficient construction of a reduced BDD for the independent set problem with bounds comparable or better than LP root node bounds (Bergman et al. 2013).

We use an IP formulation based on clique cover. Given a graph $G = (V, E)$, we generate a clique cover $\mathcal{C}$ of V. Each clique is generated by selecting a vertex with the largest degree and greedily including vertices with the largest degrees. The maximum independent set problem can be then formulated as $\max\{\sum_{v \in V} x_v : \sum_{v \in C} x_v \leq 1 \ \forall C \in \mathcal{C},\ x \in \{0,1\}^V\}$.

We test our cut generator on random graphs with varying density, using the Erdős-Rényi model. That is, we generate a graph $G(n, d)$, on n vertices, in which each pair of distinct vertices is joined by an edge with probability d. This probability d is also the average density of the graph. We examine random graphs with densities d equal to 50% and 80%. We find that both IP and BDDs tend to have more difficulty with graphs of smaller densities when the number of vertices is fixed. For this reason, the number of vertices of the graphs we test on depend on the density.

We omit experiments on graphs with lower densities since tests on 50% density instances already illustrate a case where relaxed BDDs are not tight enough for practical cuts. Lower density instances are less suitable for small tight BDD relaxations.

For experiments on gap closed and dimension, we are interested in how the strength of the cut changes as we vary the relaxation width. Thus, we use a set of smaller instances for which we can construct an exact BDD as a benchmark. We consider random graphs with 120 and 300 vertices of densities 50% and 80% respectively. For experiments on solving time and number of nodes, we consider random graphs with 250 and 400 vertices of densities 50% and 80% respectively. We examine average results for 10 random instances for each graph size and density.

We use a minimum degree variable ordering to construct the BDD: at each layer, we select the vertex with the smallest degree with respect to the vertices not yet selected.

Since an interior point is known for the maximum independent set problem, we use the interior point $\omega = (\frac{1}{2n}, \ldots, \frac{1}{2n})$, where n is the number of vertices of the instance. This is an interior point because the zero vector is feasible and all unit vectors e^j are feasible, where $e^j_i = 1$ if $i = j$ or 0 otherwise. This appears to be a reasonable choice as this is a point that is close to the origin and should not be too close to any facet except the ones defined by the nonnegative constraints. Moreover, its polyhedron is full-dimensional, so we do not have to be concerned about that aspect of cut generation.

8.1.2 Minimum set covering problem

Given a set of elements U and a collection $\mathcal{S}$ of subsets of U, the minimum set covering is to find the smallest number of sets in $\mathcal{S}$ such that their union covers all elements in U. We use the following traditional IP formulation: $\min\{\sum_{S\in\mathcal{S}} x_i : Ax \geq 1,\ x \in \{0,1\}^{\mathcal{S}}\}$, where A is such that $A_{iS} = 1$ if $i \in S$ or 0 otherwise.

We test the target cuts on set covering instances where the constraint matrix A has low bandwidth, known to support strong relaxed decision diagrams. These instances are randomly generated using the same process from Bergman et al. (2011), which creates a staircase-like structure in A. Given number of variables n, set size k, and bandwidth $b_w \geq s$, for each row i of A we select a random subset S_i of size k of $\{i, i+1, \ldots, i+b_w-1\}$, and we let $A_{ij} = 1$ if $j \in S_i$ or 0 otherwise. The number of constraints of an instance is $n - b_w + 1$.

The three groups of instances that we test on have $n = 250$ variables, set size $k = 30$, and bandwidth $b_w \in \{40, 50, 60\}$. Relaxed BDDs become weaker as the bandwidth increases, so this choice of parameters allow us to evaluate the performance of the cuts as we weaken the relaxation. We consider average results of 16 instances for each of these three groups. For the experiments on strength of cuts in which we examine the full range of widths, we only consider instances of the smallest bandwidth 40.

While a BDD can be constructed specifically for the set covering problem (Bergman et al. 2016a), in our implementation we use a generic BDD construction for linear inequalities, described in Appendix C. To ensure a strong relaxation, the variable order in the BDD is the same as the column order in the matrix A. In practice, one may run a heuristic to find an ordering that minimizes bandwidth.

Moreover, the polyhedron is full-dimensional for this particular set of instances when $k \geq 2$, as the linearly independent points $(1, \ldots, 1) - e^j$ for all $j = 1, \ldots, n$ are feasible. We use the point $(1, \ldots, 1)$ as the origin of the target cut. Although it is not an interior point, it is still a valid point if we are not concerned about generating facets of the form $x_i \leq 1$.

8.2 Experimental setup

We use CPLEX 12.6 as a MIP solver. It is commonly known that the behavior of MIP solvers can vary greatly. In order to reduce this variability and emphasize the effect of the cuts, CPLEX heuristics are disabled. We also compare our cuts with CPLEX cuts separately as we are interested in the behavior of our cuts independently from other cuts. In other words, CPLEX cuts are disabled in all of the runs we generate target cuts. The presolve reduction parameter is set to linear in all experiments, which is required for user cut generation. The root LP is solved with the barrier method for independent set and the automatic setting is used for set covering. When solving the LP to generate cuts with CPLEX, we set the preprocessing aggregation limit to a high value (100), as we find that it can substantially help in solving the LP. The experiments were performed on a 2.33Ghz Linux machine with 32GB of RAM. No issues with memory usage were observed.

The merging rule of a relaxed BDD shapes the relaxation and thus it is natural is to search for one that works well specifically for generating cuts. However, in our experience, we find that the standard merging rule that merges nodes with weak objective bounds works well for cut generation. We observe in preliminary experiments that merging rules based on Euclidean and Manhattan distance from the point to separate do not perform as well as the objective-based merging rule. In addition, we examined a rule that merges nodes based on violation with respect to a given inequality. In fact, even if we give it the inequality one would obtain with an exact BDD, we do not find a substantial improvement. Therefore, our experiments use the objective-based merging rule for both problem classes.

In our experiments, we do not attempt to obtain facet-defining cuts with a method from Section 5.4. This is because we already obtain cuts that define high-dimensional faces and the time to run the perturbation heuristic can be non-negligible for some instances. Appendix D contains further experiments using the perturbation heuristic, which show they indeed increase the face dimension fully or almost fully.

8.3 Strength of the cuts: Gap closed

Gap closed is a commonly used proxy for the effectiveness of a cut and it is defined as follows. If r is the objective value at the initial relaxation, b_k is the objective value after adding k cuts, and v^* is the optimal value of the problem, then the *gap closed* after k cuts is computed as $\frac{r-b_k}{r-v^*}$. We are also interested with these experiments in observing how strong the cuts are as we increase the maximum width of the BDD.

Figures 4 and 6 (left) display the gap closed as we iteratively generate more and more target cuts, using different widths for the relaxed BDD. The BDD width is relative to the exact width: for example, in "Width 20%" we use a relaxed BDD with 20% of the exact width, and in "Width 100%" we use an exact BDD. The graphs show that target cuts from relaxed decision diagrams indeed close a significant percentage of the gap even when the BDD is not exact. We see that they perform at its best when the relaxation is tight as expected, but the graph also shows that using a high width relaxation is often as good or almost as good as using an exact BDD. In particular, for low bandwidth set covering instances (Figure 6, left), a cut from a BDD of width 20% covers as much gap as an exact BDD. Moreover, the gap closed for the first cut is often higher than the gap closed by CPLEX at its default settings. This suggests that we can generate effective cuts with relaxed BDDs that have a reasonable width.

Additionally, the first cuts provide a much larger gap closed when compared to later cuts. The

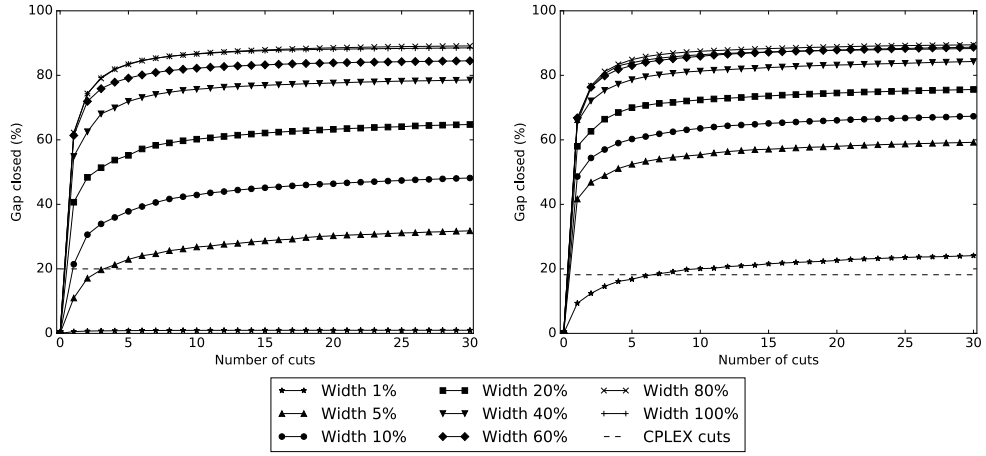

Figure 4: Gap closed for independent set instances of density 80% and 300 vertices (left) and density 50% and 120 vertices (right).

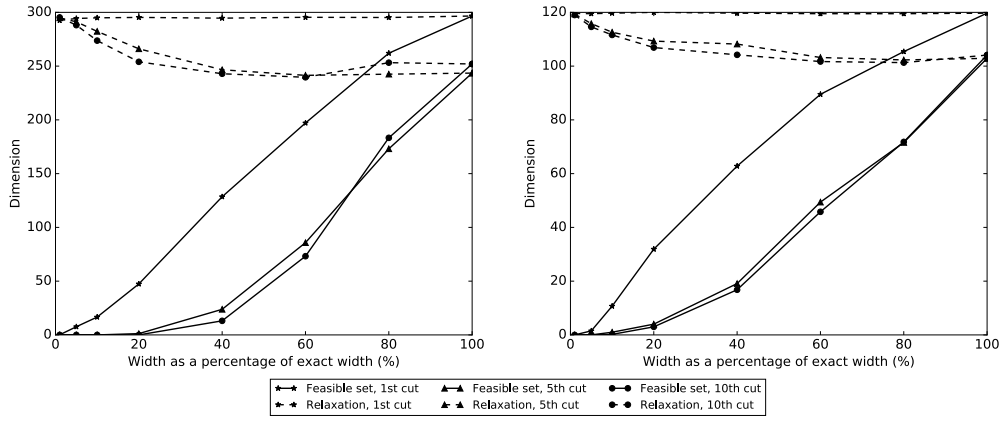

Figure 5: Face dimensions for independent set instances of density 80% and 300 vertices (left) and density 50% and 120 vertices (right).

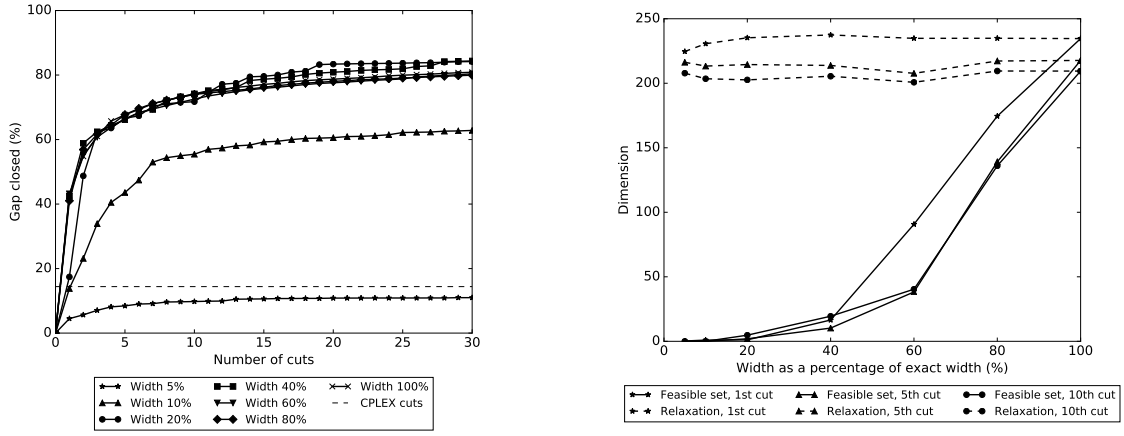

Figure 6: Gap closed (left) and face dimensions (right) for set covering instances of bandwidth 40.

observation that the initial cuts are stronger emerges in all of our experiments.

8.4 Strength of the cuts: Face dimension

A second measure of cut strength is the dimension of the face defined by a cut, with respect to the integer hull of the problem. A lower bound for these values can be obtained via flow decomposition as described in Section 6, using a violation minimizing heuristic in order to find a good set of feasible points. We numerically check if the points we obtain from the flow decomposition are affinely independent, in accordance to Theorem 5.

Figures 5 and 6 (right) exhibit these dimension bounds as the BDD width varies. We also include bounds for the BDD relaxation for context. The graphs show that increasing the width improves the dimension of the faces (of the integer hull) on these instances, which reflects the experiments on gap closed. For set covering, we do not need faces of very high dimension to close a substantial amount of gap. Additionally, we see that the initial cuts tend to have higher face dimension than subsequent cuts.

Among the instances we examined, widths of 10% for independent set and 40% for set covering were sufficient to yield nonempty feasible certificates for the first cut, meaning the cut is supporting with respect to the integer hull.

8.5 Overall performance of the cuts

In practice, we are interested in using these cuts to speed up MIP solvers. Total solving time, including BDD construction and cut generation, and number of nodes are reported in Figure 7 for the independent set problem and Figures 8 and 9 for the set covering problem. We highlight in this section independent set instances of density 80% and set covering instances of bandwidth 50 and 60, and leave graphs of the remaining instances (density 50% and bandwidth 40) to Appendix D. Appendix D also includes a breakdown on time spent in BDD construction, cut generation, and MIP solving.

For independent set, we observe that target cuts from relaxed decision diagrams are able to improve upon CPLEX with the first cut for density 80%. In particular, we obtain a significant decrease in the number of nodes and solving time as we increase the width of the relaxation for the first cut. On the other hand, for density 50% we cannot make a substantial conclusion on whether the cuts decrease the size of the tree, and the time it takes to generate the cut is not worth the realized speed up in the branch-and-bound tree.

For set covering, there is a substantial improvement in number of nodes for instances of bandwidth 40 and 50. In particular, target cuts for low-width relaxed BDDs improve solving time compared to CPLEX in these instances. For instances of bandwidth 60 however, the cuts are not effective since the relaxation is not strong enough.

8.6 Comparison with Lagrangian cuts

In this final set of experiments, we compare our cuts with a version of the Lagrangian cuts proposed by Becker et al. (2005). They are relatively fast to generate as they rely on iteratively optimizing over a BDD rather than solving a cut generating LP. The main difference is that we extract Lagrangian cuts from the same relaxed BDDs that we use to generate target cuts, while Becker et al. generate cuts from a BDD representing the feasible set of a fraction of active constraints.

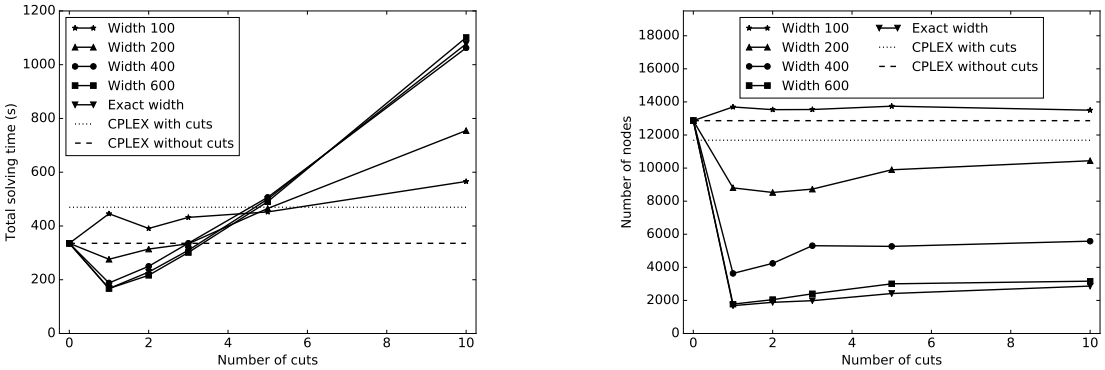

Figure 7: Solving time (left) and number of nodes of branch-and-bound tree (right) for cuts on independent set instances of density 80% with 400 vertices.

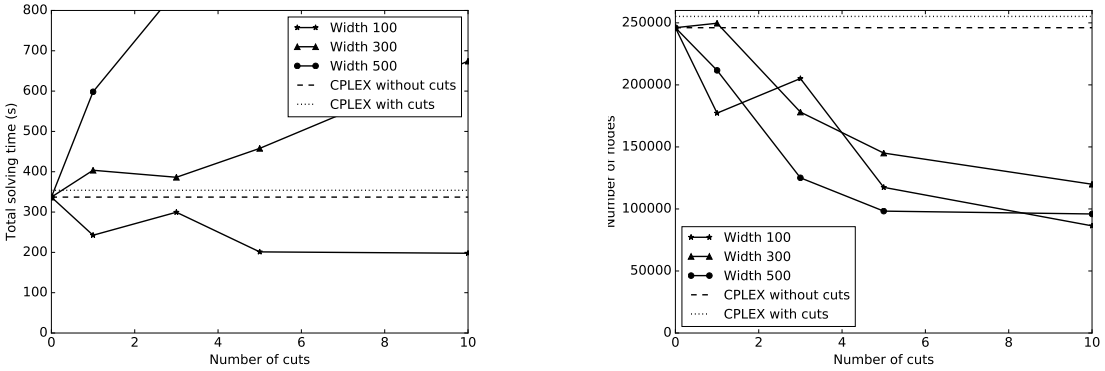

Figure 8: Solving time (left) and number of nodes of branch-and-bound tree (right) for cuts on set covering instances of bandwidth 50.

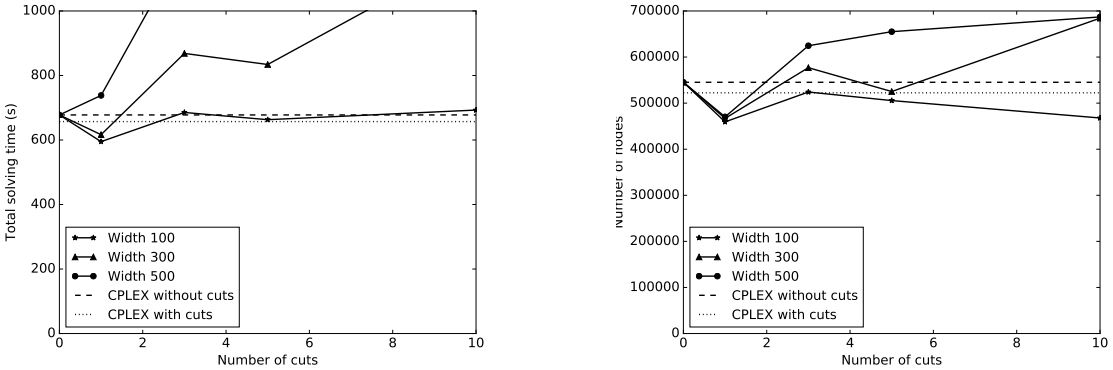

Figure 9: Solving time (left) and number of nodes of branch-and-bound tree (right) for cuts on set covering instances of bandwidth 60.

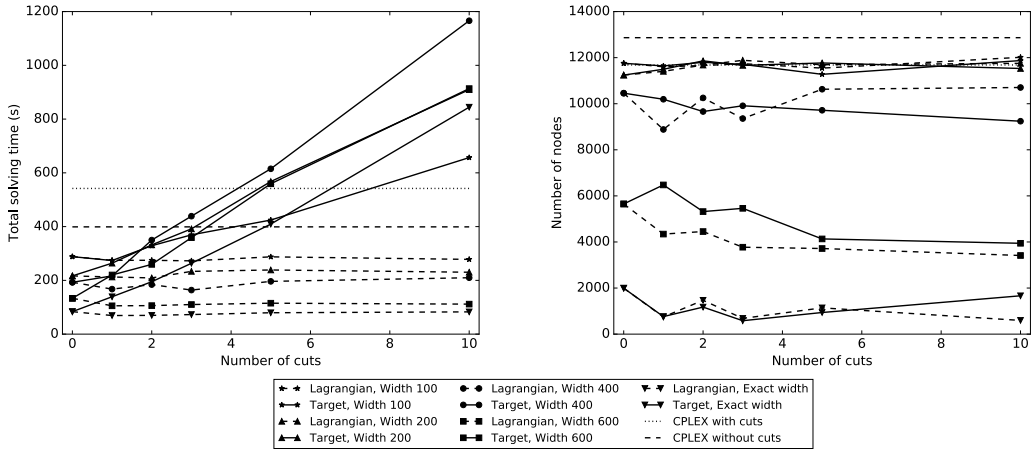

Figure 10: Comparison between target cuts and Lagrangian cuts from relaxed decision diagrams on independent set instances of density 80% and 400 vertices.

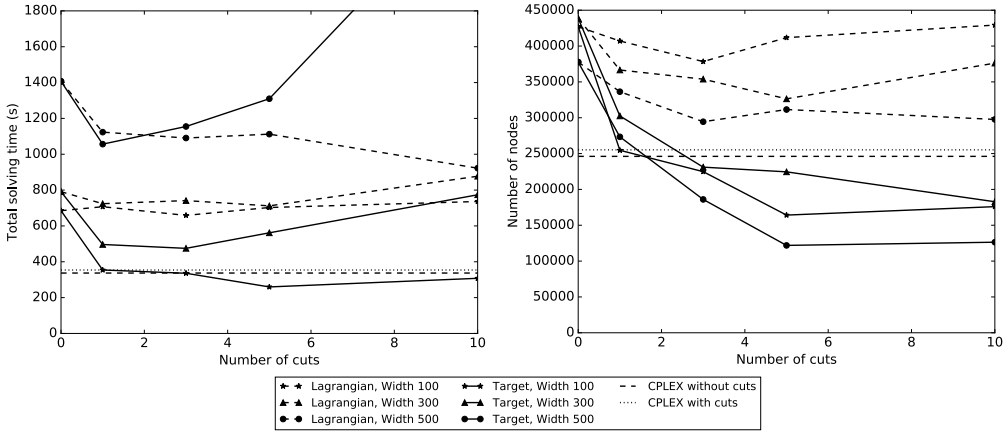

Figure 11: Comparison between target cuts and Lagrangian cuts from relaxed decision diagrams on set covering instances of bandwidth 50.

We do not apply their proposed strengthening on Lagrangian cuts since we do not do so either on target cuts. As with their version, we start with the objective as an initial solution and stop as soon as we find a valid cut.

Since the initial guess of the Lagrangian cut is the objective c, the first cut generated by the Lagrangian method is $c^\top x \leq B$ (for a maximization problem), where B is the bound given by the BDD, if B is a better bound than the root relaxation bound. This cut may be particularly helpful because it provides an upper bound for CPLEX, which can be used for pruning the tree. For a fair comparison, we add this inequality as a constraint for both cases before we start generating cuts, since it can be trivially obtained from the BDD.

For simplicity, we examine one instance group from each problem: density 80% graphs for independent set and bandwidth 50 for set covering. Figures 10 and 11 show the solving time and number of nodes in the branch-and-bound tree for target cuts and Lagrangian cuts, both with the initial bound constraint.

In the independent set case, we observe that the bound constraint from the BDD is indeed very helpful to reduce solving time. Since it is fast to generate Lagrangian cuts, it turns out that generating them is more practical in terms of total solving time than generating target cuts. We find that both target cuts and Lagrangian cuts behave similarly in terms of number of nodes, though neither provides a significant improvement with respect to the problem in which only the bound on the objective is added.

The set covering instances with low bandwidth have the unusual behavior of becoming slower if we add the bound constraint, which is why all lines start above the CPLEX lines in Figure 11. The Lagrangian cuts are not useful for these instances, likely in part because the cuts tend to be similar to the bound constraint, given that the method stops and returns a cut as soon as it is found. For both number of nodes and solving time, target cuts outperform Lagrangian cuts in these instances in most cases, despite being significantly more expensive to generate.

9 Conclusion

Previous works have shown that relaxed decision diagrams can provide good bounds, and in this work we explored an alternative viewpoint on whether they may provide good cutting planes as well. As our main contribution, we showed that generating target cuts from relaxed decision diagrams is possible due to a connection between polarity and decision diagrams.

Our experiments on the independent set problem and the set covering problem demonstrate that these cuts can be strong especially if the relaxation is tight enough. In practice, we are able to improve upon a commercial solver for certain instances, even though there is a relatively large overhead to generate the cuts.

Nevertheless, the computational results suggest that it is worthwhile to further develop these cuts and investigate their effect on other problems. We expect this cutting plane algorithm to work best on problems that have good decision diagram relaxations, but do not have tight continuous relaxations. In particular, this method can be applied to problems with nonlinear constraints as long as a suitable relaxed decision diagram is available.

Acknowledgments

The implementation used for the computational experiments in this paper is based on an existing implementation of relaxed BDDs for independent sets by Bergman et al. (2013). We greatly appreciate the improvements suggested by the anonymous reviewers of this paper.

References

Akers SB (1978) Binary decision diagrams. *IEEE Transactions on Computers* 100(6):509–516.

Andersen HR, Hadzic T, Hooker JN, Tiedemann P (2007) A constraint store based on multivalued decision diagrams. *Principles and Practice of Constraint Programming–CP 2007*, 118–132 (Springer).

Anjos MF, Liers F, Pardella G, Schmutzer A (2013) Engineering branch-and-cut algorithms for the equicut problem. *Discrete Geometry and Optimization*, 17–32 (Springer).

Becker B, Behle M, Eisenbrand F, Wimmer R (2005) BDDs in a branch and cut framework. *Experimental and Efficient Algorithms*, 452–463 (Springer).

Behle M (2007) *Binary decision diagrams and integer programming*. Ph.D. thesis, Max Planck Institute for Computer Science, Saarbrücken, Germany.

Bergman D, Cire AA, van Hoeve WJ, Hooker JN (2013) Optimization bounds from binary decision diagrams. *INFORMS Journal on Computing* 26(2):253–268.

Bergman D, Cire AA, van Hoeve WJ, Hooker JN (2016a) *Decision Diagrams for Optimization* (Springer).

Bergman D, Cire AA, van Hoeve WJ, Hooker JN (2016b) Discrete optimization with decision diagrams. *INFORMS Journal on Computing* 28(1):47–66.

Bergman D, van Hoeve WJ, Hooker JN (2011) Manipulating MDD relaxations for combinatorial optimization. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 20–35 (Springer).

Bonato T, Jünger M, Reinelt G, Rinaldi G (2014) Lifting and separation procedures for the cut polytope. *Mathematical Programming* 146(1-2):351–378.

Boyd EA (1995) On the convergence of Fenchel cutting planes in mixed-integer programming. *SIAM Journal on Optimization* 5(2):421–435.

Bryant RE (1986) Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers* 100(8):677–691.

Buchheim C, Liers F, Oswald M (2008) Local cuts revisited. *Operations Research Letters* 36(4):430–433.

Buchheim C, Liers F, Oswald M (2010) Speeding up IP-based algorithms for constrained quadratic 0–1 optimization. *Mathematical Programming* 124(1-2):513–535.

Buchheim C, Liers F, Sanità L (2011) An exact algorithm for robust network design. *Network Optimization*, 7–17 (Springer).

Cadoux F, Lemaréchal C (2013) Reflections on generating (disjunctive) cuts. *EURO Journal on Computational Optimization* 1(1-2):51–69.

Chvátal V, Cook W, Espinoza D (2013) Local cuts for mixed-integer programming. *Mathematical Programming Computation* 5(2):171–200.

Cire AA, van Hoeve WJ (2013) Multivalued decision diagrams for sequencing problems. *Operations Research* 61(6):1411–1428.

Hiriart-Urruty JB, Lemaréchal C (2001) *Fundamentals of convex analysis* (Springer).

Kell B, van Hoeve WJ (2013) An MDD approach to multidimensional bin packing. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 128–143 (Springer).

Knuth DE (2011) *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1* (Pearson Education India).

Lee CY (1959) Representation of switching circuits by binary-decision programs. *Bell System Technical Journal* 38(4):985–999.

Martin RK, Rardin RL, Campbell BA (1990) Polyhedral characterization of discrete dynamic programming. *Operations Research* 38(1):127–138.

Minato S (1993) Zero-suppressed BDDs for set manipulation in combinatorial problems. *30th Conference on Design Automation*, 272–277 (IEEE).

Schrijver A (1986) *Theory of linear and integer programming* (John Wiley & Sons).

Wegener I (2000) *Branching programs and binary decision diagrams: theory and applications*, volume 4 (SIAM).

A Algorithm to Compute the Center of Points Represented by a BDD

We present an alternative method to compute the center of S that is more suitable to our context, relative to the one proposed by Behle (2007). This may be useful to obtain an interior point for the target cut approach.

The first step is to compute, for each arc (i, j), the number of paths n_{ij} from root s to terminal t that traverse (i, j). To do so, observe that $n_{ij} = n_i^- + n_j^+$, where n_i^- is the number of paths from s to i and n_j^+ is the number of paths from j to t. Calculating the former can be done in a single top-down pass: at node s we have $n_s^- = 1$, and at each node $i \neq s$, we let $n_i^- = \sum_{j \in N^-(i)} n_j^-$, where $N^-(i)$ denotes the parents of i. Likewise, the latter can be computed in a bottom-up pass: $n_t^+ = 1$ and $n_i^+ = \sum_{j \in N^+(i)} n_j^+$, where $N^+(i)$ denotes the children of i.

We then use n_{ij} to calculate the total number $N_{\ell,k}$ of paths with label ℓ at layer k. That is, $N_{\ell,k} := \sum_{(i,j) \in S_{\ell,k}} n_{ij}$, where $S_{\ell,k}$ is the set of ℓ-arcs in layer k. Since paths of B correspond to points of S, $N_{\ell,k}$ is the total number of points $x \in S$ such that $x_k = \ell$. In addition, note that $N := \sum_{(i,j) \in \delta^+(s)} n_{ij}$ is the total number of points of S. Therefore, the center of S can be expressed as $\frac{1}{N}(0N_{0,1} + 1N_{1,1}, \ldots, 0N_{0,n} + 1N_{1,n}) = \frac{1}{N}(N_{1,1}, \ldots, N_{1,n})$.

B Proof of Theorem 5 on Face Certificates

We first restate Theorem 5, which shows how to obtain a face certificate for a target cut from a decision diagram. All definitions required for the theorem, such as (P), (D), $\tilde{P}_{\text{flow}}$, and $S^+_{\alpha^*}$, can be found in Section 6.

Theorem 5. *Let (u^*, v^*) and f^* be an optimal primal-dual pair for (P) and (D). If f^* is an extreme point of $\tilde{P}_{flow}$ and α^* is the result of a standard flow decomposition applied to f^*, then $S^+_{\alpha^*}$ is a $|S^+_{\alpha^*}|$-face certificate for $u^{*\top} x \leq 1$ with respect to conv(S).*

We prove a series of intermediate results and the theorem will follow. For simplicity, throughout this section we assume the origin is in the interior of $\text{conv}(S)$. At the end of this section, we argue that all of the following results still hold without this assumption.

We put aside decision diagrams for a moment and briefly discuss how to obtain certificates if we had the set of points S explicitly. In this scenario, target cuts may be generated by solving $\max_u\{u^\top \bar{x} : u \in S^\circ\}$, denoted by ($\tilde{\text{P}}$). Its dual is $\min_\alpha\{\sum_{x \in S} \alpha_x : \alpha \in P_{\text{cone}}\}$, denoted by ($\tilde{\text{D}}$), where

$$P_{\text{cone}} = \{\alpha : \tilde{S}\alpha = \bar{x},\ \alpha \geq 0\}$$

and $\tilde{S}$ is the matrix formed by the points of S in its columns. In other words, P_{cone} is the polyhedron of the coefficients of the conic combinations that express $\bar{x}$.

In the following proposition, we view a basis J of $\tilde{S}$ as a set of points of S, since $\tilde{S}$ represents a set of points of S in its columns. Moreover, we call a $\dim(S)$-face certificate a *facet certificate*.

Proposition 2. *Let u^* and α^* be an optimal primal-dual pair for ($\tilde{P}$) and ($\tilde{D}$).*

(i) If J is an optimal basis for α^*, then J is a facet certificate for $u^{*\top} x \leq 1$ with respect to conv(S).

(ii) If α^* is an extreme point of P_{cone}, then $S^+_{\alpha^*}$ is a $|S^+_{\alpha^*}|$-face certificate for $u^{*\top}x \leq 1$ with respect to conv(S).

Proof. Since basic variables have zero reduced costs, $x \in J$ implies $u^{*\top}x = 1$. Therefore, all points in J are tight with respect to $u^{*\top}x \leq 1$. Moreover, J is a maximal linearly independent set by definition, and thus J is a facet certificate.

If we do not have an optimal basis but α^* is an extreme point of P_{cone}, then there exists a basis associated to α^* that contains $S^+_{\alpha^*}$. Therefore, all points in $S^+_{\alpha^*}$ are tight with respect to $u^{*\top}x \leq 1$ and are linearly independent, implying $S^+_{\alpha^*}$ is a $|S^+_{\alpha^*}|$-face certificate. □

The weaker result (ii) from Proposition 2 is useful to prove Theorem 5. In the decision diagram case, we cannot obtain a facet certificate from (P) in general without modifying its optimal u^* because u^* is not guaranteed to define a facet without additional steps, as discussed in Section 5.4.

Our goal now is to derive a similar result for the case where S is implicitly represented as a decision diagram. That is, instead of ($\tilde{\text{P}}$), we solve (P). Recall that the dual (D) of (P) is $\min_f\{\sum_{j\in\delta^+(s)} f_{sj} : f \in \tilde{P}_{\text{flow}}\}$. In this section, it is convenient to express the constraints of $\tilde{P}_{\text{flow}}$ using matrix notation:

$$\tilde{P}_{\text{flow}} = \{f : Nf = 0,\ Vf = \bar{x},\ f \geq 0\}.$$

Here, N is the node-arc incidence matrix and V is the arc value matrix. That is, N_{ie} is 1 if arc e incides on vertex i and 0 otherwise and V_{ke} is the value that arc e assigns to variable x_k.

In order to link $\tilde{P}_{\text{flow}}$ to P_{cone}, we rely on the following polyhedron that maps a fixed $f \in \tilde{P}_{\text{flow}}$ to $\alpha \in P_{\text{cone}}$:

$$P_{\text{dec}}(f) = \{\alpha : H\alpha = f, \alpha \geq 0\},$$

where H is the arc-path incidence matrix, that is, H_{ep} is 1 if arc e is in path p or 0 otherwise. This polyhedron represents all possible ways that a fixed $f \in \tilde{P}_{\text{flow}}$ can be decomposed into $\alpha \in P_{\text{cone}}$, which is shown in the next proposition.

Proposition 3. *If $\alpha \in P_{cone}$, then $H\alpha \in \tilde{P}_{flow}$. Conversely, if $f \in \tilde{P}_{flow}$ and $\alpha \in P_{dec}(f)$, then $\alpha \in P_{cone}$.*

Proof. First note that all entries in the NH matrix are zero: each node has a single incoming arc and a single outgoing arc in a path, which cancel out when calculating NH. In addition, $VH = \tilde{S}$, since $(VH)_{kp}$ corresponds to the value assigned to x_k by the path p.

Let $\alpha \in P_{\text{cone}}$. Then $N(H\alpha) = 0$, $V(H\alpha) = \tilde{S}\alpha = \bar{x}$, and $H\alpha \geq 0$. Therefore, $H\alpha \in \tilde{P}_{\text{flow}}$.

Let $f \in \tilde{P}_{\text{flow}}$ and $\alpha \in P_{\text{dec}}(f)$. Then $\tilde{S}\alpha = VH\alpha = Vf = \bar{x}$ and $\alpha \geq 0$. Therefore, $\alpha \in P_{\text{cone}}$. □

Next, we prove two lemmas that together imply Theorem 5.

Lemma 1. *Let α be the result of a standard flow decomposition applied to $f \in \tilde{P}_{flow}$. Then α is an extreme point of $P_{dec}(f)$.*

Proof. It suffices to show that the columns x of $\tilde{S}$ such that $\alpha_x > 0$ are linearly independent. The reason is that, if so, there must exist a basis J of $\tilde{S}$ containing these columns. Moreover, J must

be associated to α since $\tilde{S}_J \alpha_J = \tilde{S}\alpha = \bar{x}$, given that $\alpha_x = 0$ for all $x \notin J$. This implies α is an extreme point.

Let t be the number of iterations of the decomposition, or equivalently the number of points x with $\alpha_x > 0$ at the end of the decomposition. Consider a $t \times t$ matrix M where the columns (paths) are the ones with positive flow from the decomposition in the order they were encountered, and the rows (arcs) are the bottleneck arcs of each of those paths also in the same order. Then M is lower triangular since a bottleneck arc never reappears in a path after its flow is set to zero. Therefore, these columns are linearly independent. □

The following result connects extreme points of $\tilde{P}_{\text{flow}}$ and $P_{\text{dec}}(f)$ with extreme points of P_{cone}.

Lemma 2. *If f is an extreme point of $\tilde{P}_{flow}$ and α is an extreme point of $P_{dec}(f)$, then α is an extreme point of P_{cone}.*

Proof. Suppose for contradiction there exist $\alpha^1, \alpha^2 \in P_\alpha$ such that $\alpha = \lambda\alpha^1 + (1-\lambda)\alpha^2$ with $\lambda > 0$. We show that if α^1 and α^2 correspond to different flows, then f is not an extreme point of $\tilde{P}_{\text{flow}}$, and if they correspond to the same flow, then α is not an extreme point of $P_{\text{dec}}(f)$.

Case 1: $H\alpha^1 \neq H\alpha^2$. Define $f^t = H\alpha^t$ for $t = 1, 2$. By Proposition 3, $f^t \in \tilde{P}_{\text{flow}}$. Moreover, $f = H\alpha = H(\lambda\alpha^1 + (1-\lambda)\alpha^2) = \lambda H\alpha^1 + (1-\lambda)H\alpha^2 = \lambda f^1 + (1-\lambda)f^2$, with f^1 and f^2 distinct. Thus f is not an extreme point of $\tilde{P}_{\text{flow}}$, a contradiction.

Case 2: $H\alpha^1 = H\alpha^2$. It suffices to show that $\alpha^1, \alpha^2 \in P_{\text{dec}}(f)$, thus proving that α is not an extreme point of $P_{\text{dec}}(f)$. We have $f = H\alpha = H(\lambda\alpha^1 + (1-\lambda)\alpha^2) = \lambda(H\alpha^1 - H\alpha^2) + H\alpha^2 = H\alpha^2$. This also implies $f = H\alpha^1$. Therefore, $\alpha^1, \alpha^2 \in P_{\text{dec}}(f)$. □

Finally, we complete the proof of Theorem 5 by putting together the previous results.

Proof. Proof of Theorem 5. By Lemmas 1 and 2, α^* is an extreme point of P_{cone}. The result then follows from Proposition 2. □

As discussed in Section 5.4, the cut generation algorithm may involve translating points if the origin is not in the interior of $\text{conv}(S)$. While this may take away the linear independence of the points of S^+_α, they remain affinely independent, and thus all previous results still hold.

C Decision Diagram Construction for Linear Inequalities

For the set covering problem, we use a generic BDD construction for linear inequalities. We provide a brief description of this construction in this section.

Given a set of m constraints $a_i^\top x \leq b_i$, each state carries m values representing the right-hand sides of each constraint after assigning a partial solution. More precisely, a state is a vector $s \in \mathbb{R}^m$ where $s_i = b_i - \sum_{j \in S_p} a_{ij} x_j$ and S_p is the set of variable indices that are already assigned at the state. Therefore, the transition function subtracts a_{ij} from s_i if x_j is assigned to one, and s_i is kept unchanged if x_j is assigned to zero. Merging non-equivalent states into a relaxed one is done by keeping the most relaxed right-hand side for each constraint.

In addition, we maintain the range that the left-hand side can take for each constraint in a state. This allows us to perform a number of simple checks in order to remove or merge nodes more often. More specifically, they are used to verify if a constraint is infeasible or is feasible for

all remaining assignments. They are also used in a simple propagation scheme across constraints within a state.

With these additional checks, this construction applied to the set covering formulation turns out to be equivalent to the one specific to the set covering problem described by Bergman et al. (2016a, Section 3.6), except that its genericness adds an overhead of time and memory.

D Additional Computational Results

In this section, we present additional graphs on computational results. Figure 12 depicts solving time and number of nodes for independent set instances of density 50%, in which we see target cuts have little effect due to the relatively weak BDD relaxation. Figure 13 shows solving time and number of nodes for set covering instances of bandwidth 40, which are similar to the results for bandwidth 50.

Figure 14 gives us a breakdown of where the time is spent for the first cut for independent set instances. In both situations, the time to construct a decision diagram is very small compared to the time to solve the LP to generate a cut – less than 2%. For density 80%, we can observe that it is not too time consuming to generate a cut and it significantly reduces the time spent in the branch-and-bound tree. This however does not happen for density 50%, where generating a cut is significantly more expensive than solving the root LP relaxation. We also see in density 80% how increasing the width of the BDD yields a stronger cut that significantly reduces the time spent in the branch-and-bound tree.

Figure 15 shows the same graphs from Figure 5, in which we examined the dimension of the faces defined by the cuts, except that in this case we apply the perturbation heuristic. If we compare them to Figure 5, we observe that the perturbation heuristic does as expected: it fully or almost fully increases the dimensions of the cuts with respect to the relaxation.

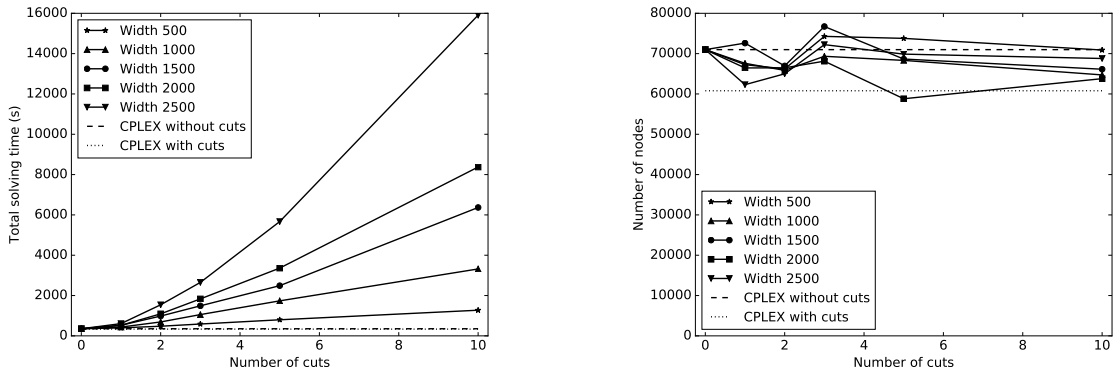

Figure 12: Solving time (left) and number of nodes of branch-and-bound tree (right) for cuts on independent set instances of density 50% with 250 vertices.

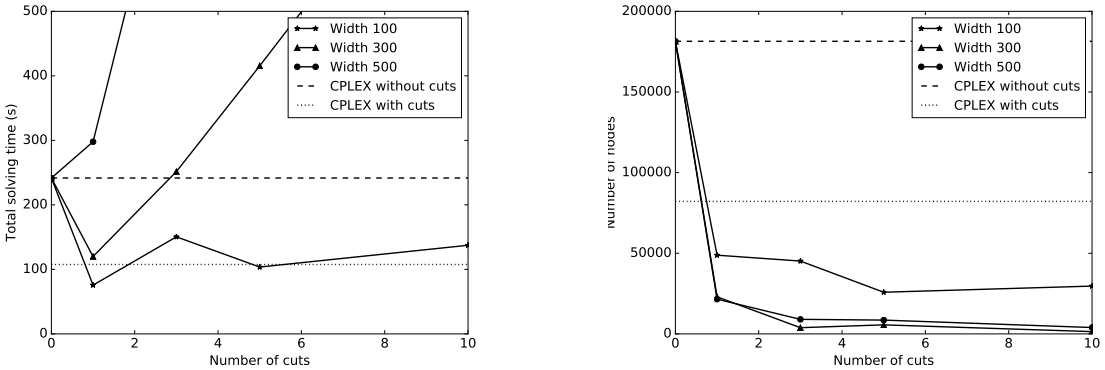

Figure 13: Solving time (left) and number of nodes of branch-and-bound tree (right) for cuts on set covering instances of bandwidth 40.

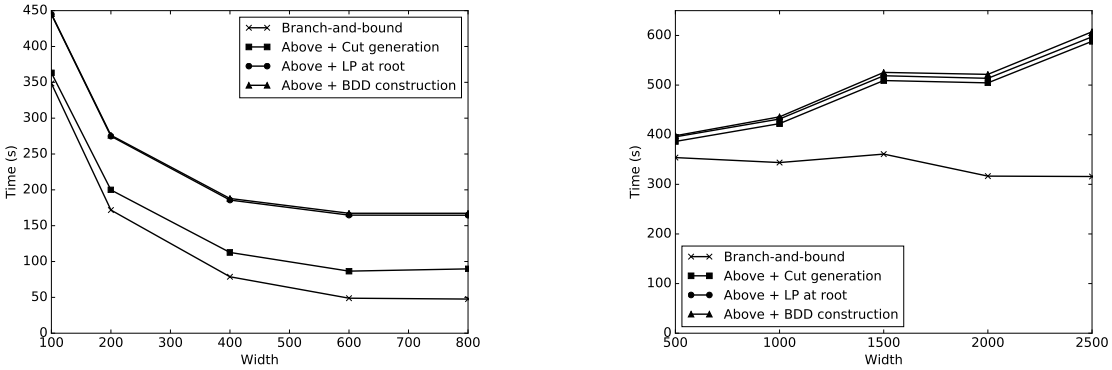

Figure 14: Total solving time breakdown with a single cut for independent set instances of density 80% with 400 vertices (left) and 50% with 250 vertices (right).

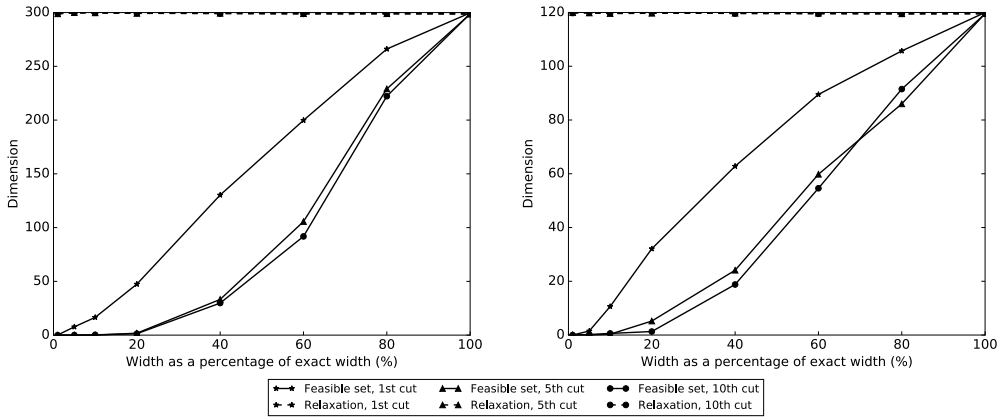

Figure 15: Face dimensions for independent set instances of density 80% and 300 vertices (left) and density 50% and 120 vertices (right), after applying the perturbation heuristic to increase dimension.